Module Specification

Principles of Programming

Version: 2026-27, v2.0, Approved

Contents

Module Specification **1**

Part 1: Information **2**

Part 2: Description **2**

Part 3: Teaching and learning methods **4**

Part 4: Assessment **5**

Part 5: Contributes towards **6**

Part 1: Information

Module title: Principles of Programming

Module code: UFCFHS-30-1

Level: Level 4

For implementation from: 2026-27

UWE credit rating: 30

ECTS credit rating: 15

College: College of Arts, Technology and Environment

School: CATE School of Computing and Creative Technologies

Partner institutions: None

Field: Computer Science and Creative Technologies

Module type: Module

Pre-requisites: None

Excluded combinations: None

Co-requisites: None

Continuing professional development: No

Professional, statutory or regulatory body requirements: None

Part 2: Description

Overview: This module introduces the basic programming concepts, techniques, and fundamental problem solving and software development principles. Topics include variables and assignment, primitive and complex data types, control structures, iteration, functions, recursion, arrays, pointers, memory management. It also introduces programming paradigms, for instance, fundamental ideas and principles behind the object-oriented approach to programming, including classes, objects, encapsulation, abstraction, inheritance, and polymorphism. The module also

teaches the basics of functional programming and introduces basic software development topics such as design, testing, and debugging. It also addresses aspects like vulnerabilities, defects and bugs in code so that students are aware of potential threats and attacks. This module suits for both the absolute beginners and those who have no prior programming experience.

Features: Not applicable

Educational aims: Through a practice-led approach, this module aims to give students a solid foundational knowledge and understanding of how programming languages are used to develop software.

Outline syllabus: Topics are likely to include but are not limited to:

Introduction to Imperative Programming
Core imperative programming concepts: sequences, selection, iteration, recursion, assignment, constants and variables. Data types, Functions, Parameters and scope of variables, Arrays, Pointers and memory management, File I/O

Program design
Problem solving and program design with pseudo code (thinking algorithmically). Developing small programs using functions. Designing and implementing simple algorithms and data structures, algorithm or program complexity (Big-O notations). Basic software development techniques: Implementation, debugging, testing, code review and version control using standard IDEs such as NetBeans or Eclipse

Functional programming
Programming with a functional notation, sessions and scripts, lambda functions, call-back functions, expressions and values. Evaluation of expressions, Lists and their operations, Recursion and induction.

Object-Oriented Principles
Fundamental principles and concepts of object-oriented programming; Abstract data types; Class Anatomy and its object creation; Encapsulation; Overloading principle; Object interaction; Abstract classes and Interfaces; Inheritance; Basic polymorphism;

Overriding principle; Static and Dynamic Binding; Exception handling; Design, code and test a series of object-oriented programs.

Part 3: Teaching and learning methods

Teaching and learning methods: This module will principally be delivered as combination of lectures and practical sessions with some occasional tutorials and seminars. Students are expected to attend all scheduled classes. We encourage students to be active in their learning. We provide a range of resources and activities to enable them to engage in achieving the learning outcomes.
The lectures will explain theoretical concepts. The theory will be underpinned by practical sessions during which the students will solve problems and write and experiment with programme code to implement those solutions.
The module will be supported by the University's Virtual Learning Environment which will be used as a repository for course materials, a forum for discussion and, from time to time, tests and/or quizzes to enable the students to self-test their knowledge.

Module Learning outcomes: On successful completion of this module students will achieve the following learning outcomes.

MO1 Apply programming constructs and algorithmic principles to design, implement, and evaluate solutions to computational problems, reflecting professional software development practices.

MO2 Develop and express solutions algorithmically and programmatically to solve simple problems.

MO3 Implement and evaluate solutions using different programming paradigms (for example functional or object oriented) identifying potential code vulnerabilities.

MO4 Apply good programming style and secure coding practices, and interpret their impact on developing, performance, testing, and debugging programs.

Hours to be allocated: 300

Contact hours:

Independent study/self-guided study = 228 hours

Face-to-face learning = 72 hours

Reading list: The reading list for this module can be accessed at readinglists.uwe.ac.uk via the following link https://rl.talis.com/3/uwe/lists/5A97CB0E-403B-A407-5F43-2C427D494983.html

Part 4: Assessment

Assessment strategy: The assessment on this module has been designed to build the students' confidence in writing software. The first assessment is a two in-class tests of increasing difficulty. Taken together, the tests form the mark for this assessment. The tests are provided through an automated system that provides realtime feedback and indicates their understanding of topics presented.

As a controlled condition assessment it is also designed to help the students build their confidence in the assessment process. For this early module, rather than running a formal exam the controlled condition assessment in both the main sit and the resit, takes place in the classroom setting.

The second assessment is an individual coursework assignment of problem solving, algorithm design and program implementation including debugging and testing. It will be an assignment portfolio submission including, design, coding and testing documents and an in-class software demonstration. The resit for this assessment will be design, coding and testing of a software system submitted as an individual portfolio with supporting software.

Students will have the opportunity for formative feedback during practical lab/tutorial sessions.

Assessment tasks:

In-class test (First Sit)
Description: In class programming tests split into two parts (sat on different dates).

Weighting: 50 %
Final assessment: No
Group work: No
Learning outcomes tested: MO2, MO3

Portfolio (First Sit)
Description: Individual software development coursework submitted online, assessed via a 10-minute in-class or video demonstration.
Weighting: 50 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO1, MO3, MO4

In-class test (Resit)
Description: In class programming tests (sat on one day).
Weighting: 50 %
Final assessment: No
Group work: No
Learning outcomes tested: MO2, MO3

Portfolio (Resit)
Description: Individual software development coursework submitted online, assessed via a 10-minute in-class or video demonstration.
Weighting: 50 %
Final assessment: Yes
Group work: No
Learning outcomes tested: MO1, MO3, MO4

Part 5: Contributes towards

This module contributes towards the following programmes of study:

Computer Science {Foundation} [GCET] BSc (Hons) 2025-26

Computer Science (Artificial Intelligence) {Foundation} [GCET] BSc (Hons) 2025-26

Computer Science {Foundation} [GCET] DipHE 2025-26

Computer Science (Artificial Intelligence) {Foundation} [GCET] DipHE 2025-26

Computer Science (Smart Devices) {Foundation} [GCET] DipHE 2025-26

Computer Science (Smart Devices) {Foundation} [GCET] BSc (Hons) 2025-26

Computer Science {Foundation} [Frenchay] BSc (Hons) 2025-26

Computer Science [Frenchay] BSc (Hons) 2026-27

Computer Science {Dual} [Taylors] BSc (Hons) 2026-27

Computer Science [Frenchay] BSc (Hons) 2026-27

Computer Science {Foundation} [Frenchay] BSc (Hons) 2025-26

Computer Science (Artificial Intelligence) [NepalBrit] BSc (Hons) 2026-27

Computer Science [Villa] BSc (Hons) 2026-27

Computer Science [Phenikaa] BSc (Hons) 2026-27

Computer Science [Villa] BSc (Hons) 2026-27

Computer Science [Phenikaa] BSc (Hons) 2026-27